

# Predicting Daily Land Average Temperature

## Load the Data

Load the daily land-surface average anomaly data provided by Berkeley Earth collected from 1880 to 2022.

```
In [1]: import pandas as pd

url = "https://berkeley-earth-temperature.s3.us-west-1.amazonaws.com/Global/
...
read the data from the url link
ignore the comments starting with '%'
ignore the header in the comments and assign manually
...
df = pd.read_csv(url, sep=r"\s+", comment="#", header=None)

# assign column headers
column_names = ["Date Number", "Year", "Month", "Day", "Day of Year", "Anoma
df.columns = column_names

# df.to_csv("../data/raw.csv", index=False)
```

```
In [2]: df
```

```
Out[2]:
```

	Date Number	Year	Month	Day	Day of Year	Anomaly
0	1880.001	1880	1	1	1	-0.692
1	1880.004	1880	1	2	2	-0.592
2	1880.007	1880	1	3	3	-0.673
3	1880.010	1880	1	4	4	-0.615
4	1880.012	1880	1	5	5	-0.681
...	...	...	...	...	...	...
52072	2022.568	2022	7	27	208	1.639
52073	2022.571	2022	7	28	209	1.631
52074	2022.574	2022	7	29	210	1.574
52075	2022.577	2022	7	30	211	1.577
52076	2022.579	2022	7	31	212	1.629

52077 rows × 6 columns

# Data Preprocessing

```
In [3]: df.isna().sum()
```

```
Out[3]: Date Number      0  
Year          0  
Month         0  
Day           0  
Day of Year   0  
Anomaly        0  
dtype: int64
```

```
In [4]: df.dtypes
```

```
Out[4]: Date Number      float64  
Year          int64  
Month         int64  
Day           int64  
Day of Year   int64  
Anomaly        float64  
dtype: object
```

```
In [5]: # df = df.drop(columns=['Date Number'])
```

```
In [6]: BASELINE_TEMP = 8.59 # Jan 1951–Dec 1980 land-average temperature in celsius  
df['Temperature'] = df['Anomaly'] + BASELINE_TEMP
```

```
In [7]: month_dict = {  
    1: 'January',  
    2: 'February',  
    3: 'March',  
    4: 'April',  
    5: 'May',  
    6: 'June',  
    7: 'July',  
    8: 'August',  
    9: 'September',  
    10: 'October',  
    11: 'November',  
    12: 'December'  
}  
  
df['Month_Name'] = df['Month'].map(month_dict)
```

```
In [8]: df
```

Out[8]:

	Date Number	Year	Month	Day	Day of Year	Anomaly	Temperature	Month_Name
0	1880.001	1880	1	1	1	-0.692	7.898	January
1	1880.004	1880	1	2	2	-0.592	7.998	January
2	1880.007	1880	1	3	3	-0.673	7.917	January
3	1880.010	1880	1	4	4	-0.615	7.975	January
4	1880.012	1880	1	5	5	-0.681	7.909	January
...	...	...	...	...	...	...	...	...
52072	2022.568	2022	7	27	208	1.639	10.229	July
52073	2022.571	2022	7	28	209	1.631	10.221	July
52074	2022.574	2022	7	29	210	1.574	10.164	July
52075	2022.577	2022	7	30	211	1.577	10.167	July
52076	2022.579	2022	7	31	212	1.629	10.219	July

52077 rows × 8 columns

## Exploratory Data Analysis (EDA)

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 52077 entries, 0 to 52076
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   Date Number  52077 non-null   float64
 1   Year         52077 non-null   int64  
 2   Month        52077 non-null   int64  
 3   Day          52077 non-null   int64  
 4   Day of Year  52077 non-null   int64  
 5   Anomaly      52077 non-null   float64
 6   Temperature  52077 non-null   float64
 7   Month_Name   52077 non-null   object 
dtypes: float64(3), int64(4), object(1)
memory usage: 3.2+ MB
```

In [10]:

```
df.describe()
```

Out[10]:

	Date Number	Year	Month	Day	Day of Year	
<b>count</b>	52077.000000	52077.000000	52077.000000	52077.000000	52077.000000	5207
<b>mean</b>	1951.290840	1950.791693	6.512779	15.729228	182.688577	1
<b>std</b>	41.160006	41.160470	3.447762	8.799991	105.302088	
<b>min</b>	1880.001000	1880.000000	1.000000	1.000000	1.000000	-
<b>25%</b>	1915.648000	1915.000000	4.000000	8.000000	92.000000	-1
<b>50%</b>	1951.292000	1951.000000	7.000000	16.000000	183.000000	
<b>75%</b>	1986.936000	1986.000000	10.000000	23.000000	274.000000	1
<b>max</b>	2022.579000	2022.000000	12.000000	31.000000	365.000000	

In [11]: `df.isna().any()`

Out[11]:

Date Number	False
Year	False
Month	False
Day	False
Day of Year	False
Anomaly	False
Temperature	False
Month_Name	False
dtype: bool	

In [12]: `# month order from above dict values  
month_order = list(month_dict.values())  
month_order`

Out[12]:

```
[ 'January',
  'February',
  'March',
  'April',
  'May',
  'June',
  'July',
  'August',
  'September',
  'October',
  'November',
  'December']
```

In [13]: `# Order the month names as categorical  
df['Month_Name'] = pd.Categorical(df['Month_Name'], categories=month_order,  
df['Month_Name'])`

```
Out[13]: 0      January
1      January
2      January
3      January
4      January
...
52072    July
52073    July
52074    July
52075    July
52076    July
Name: Month_Name, Length: 52077, dtype: category
Categories (12, object): ['January' < 'February' < 'March' < 'April' ... 'September' < 'October' < 'November' < 'December']
```

```
In [14]: # Split into test and train dataframes based on cutoff
# beyond which we would like to score our model and use on unseen examples
cutoff = 2018
test_df = df[df['Year']>=cutoff]
test_df
```

```
Out[14]:
```

	Date Number	Year	Month	Day	Day of Year	Anomaly	Temperature	Month_Name
50404	2018.001	2018	1	1	1	1.194	9.784	January
50405	2018.004	2018	1	2	2	1.314	9.904	January
50406	2018.007	2018	1	3	3	1.358	9.948	January
50407	2018.010	2018	1	4	4	1.390	9.980	January
50408	2018.012	2018	1	5	5	1.517	10.107	January
...	...	...	...	...	...	...	...	...
52072	2022.568	2022	7	27	208	1.639	10.229	July
52073	2022.571	2022	7	28	209	1.631	10.221	July
52074	2022.574	2022	7	29	210	1.574	10.164	July
52075	2022.577	2022	7	30	211	1.577	10.167	July
52076	2022.579	2022	7	31	212	1.629	10.219	July

1673 rows × 8 columns

```
In [15]: train_df = df[df['Year']<cutoff]
train_df
```

Out[15]:

	Date Number	Year	Month	Day	Day of Year	Anomaly	Temperature	Month_Name
0	1880.001	1880	1	1	1	-0.692	7.898	January
1	1880.004	1880	1	2	2	-0.592	7.998	January
2	1880.007	1880	1	3	3	-0.673	7.917	January
3	1880.010	1880	1	4	4	-0.615	7.975	January
4	1880.012	1880	1	5	5	-0.681	7.909	January
...	...	...	...	...	...	...	...	...
50399	2017.988	2017	12	27	361	1.046	9.636	December
50400	2017.990	2017	12	28	362	1.445	10.035	December
50401	2017.993	2017	12	29	363	1.594	10.184	December
50402	2017.996	2017	12	30	364	1.506	10.096	December
50403	2017.999	2017	12	31	365	1.302	9.892	December

50404 rows × 8 columns

In [16]:

```
import altair as alt
alt.data_transformers.enable('vegafusion')

temp_plot = alt.Chart(train_df).mark_line().encode(
    x = 'Year:T',
    y = 'Temperature:Q'
).properties(width = 700, height = 400)

temp_plot

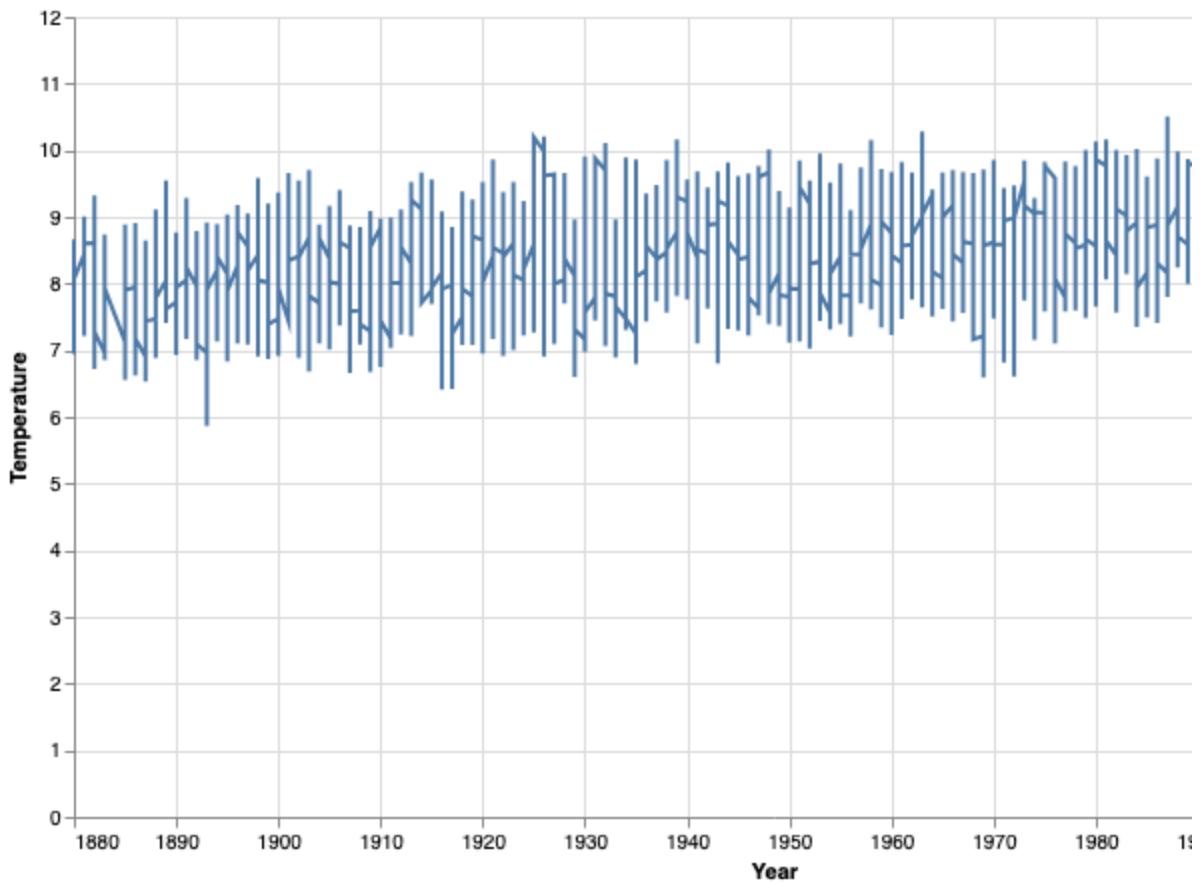
# Suffering from Overplotting, perhaps we should take the mean of the temper
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?

```
    return _is_pandas_dataframe(obj) or isinstance(
```

Out[16]:



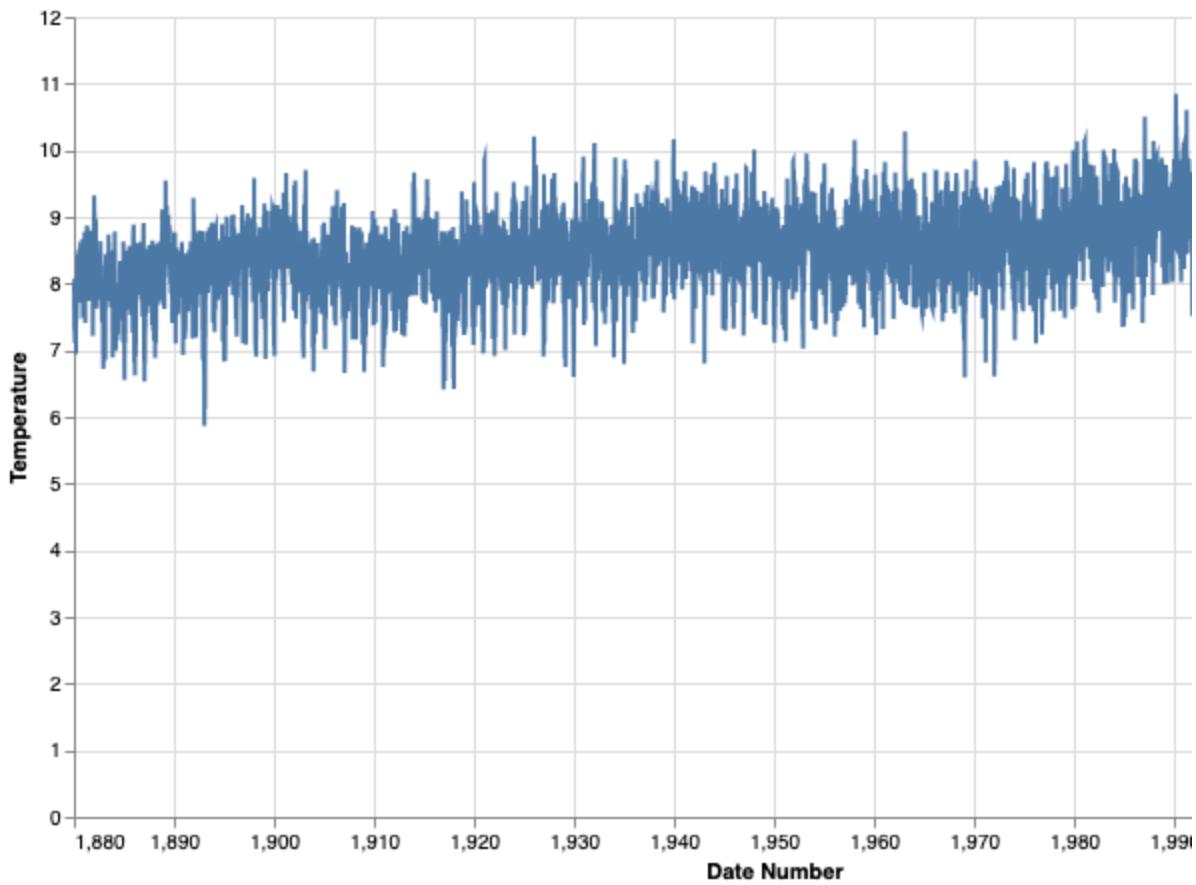
```
In [17]: temp_plot = alt.Chart(train_df).mark_line().encode(
    x = 'Date Number:Q',
    y = 'Temperature:Q'
).properties(width = 700, height = 400)

temp_plot
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?  
return \_is\_pandas\_dataframe(obj) or isinstance(

Out[17]:



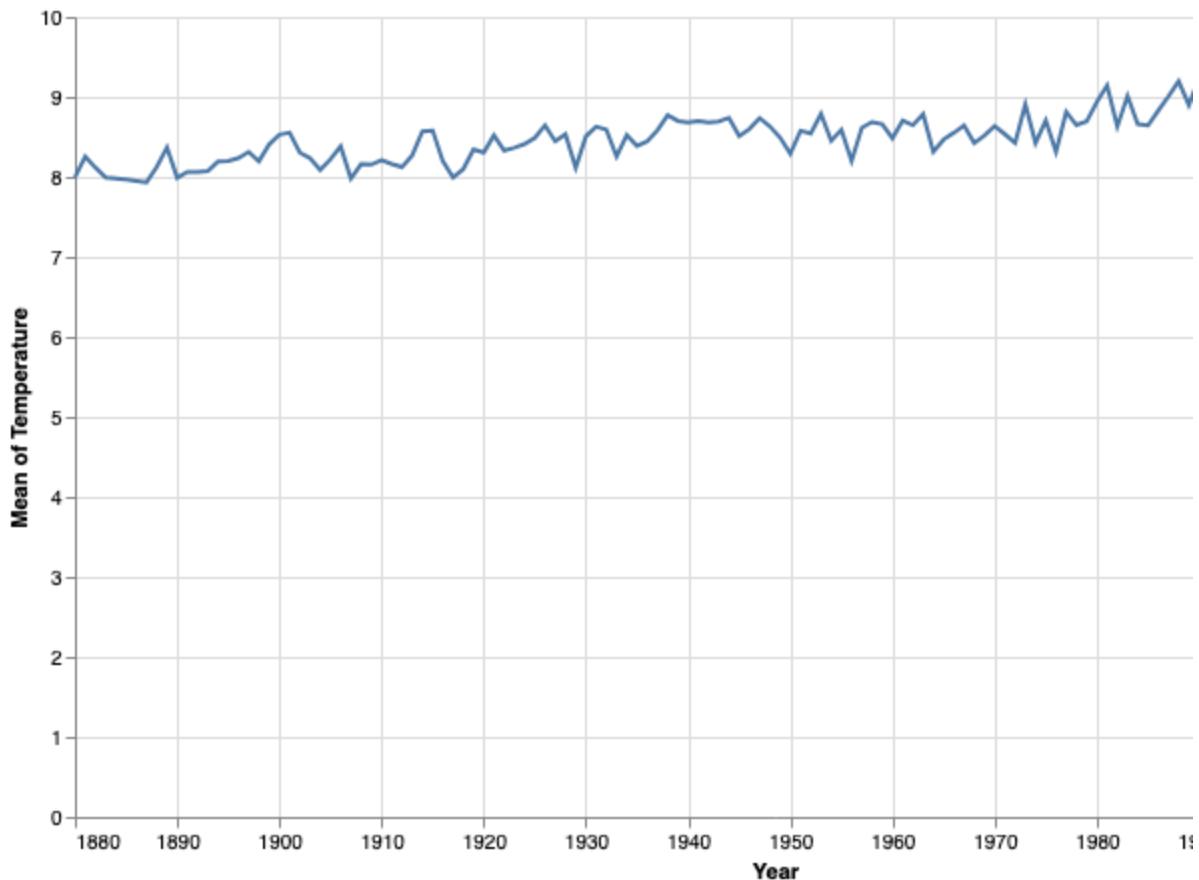
```
In [18]: temp_plot = alt.Chart(train_df).mark_line().encode(
    x = 'Year:T',
    y = 'mean(Temperature)'
).properties(width = 700, height = 400).facet('Month', columns=2)

temp_plot
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?  
return \_is\_pandas\_dataframe(obj) or isinstance(

Out[18]:



```
In [19]: temp_plot = alt.Chart(train_df).mark_line().encode(
    x = 'Year:T',
    y = 'Temperature:Q'
).properties(width = 400, height = 200).facet('Month_Name', columns=2)

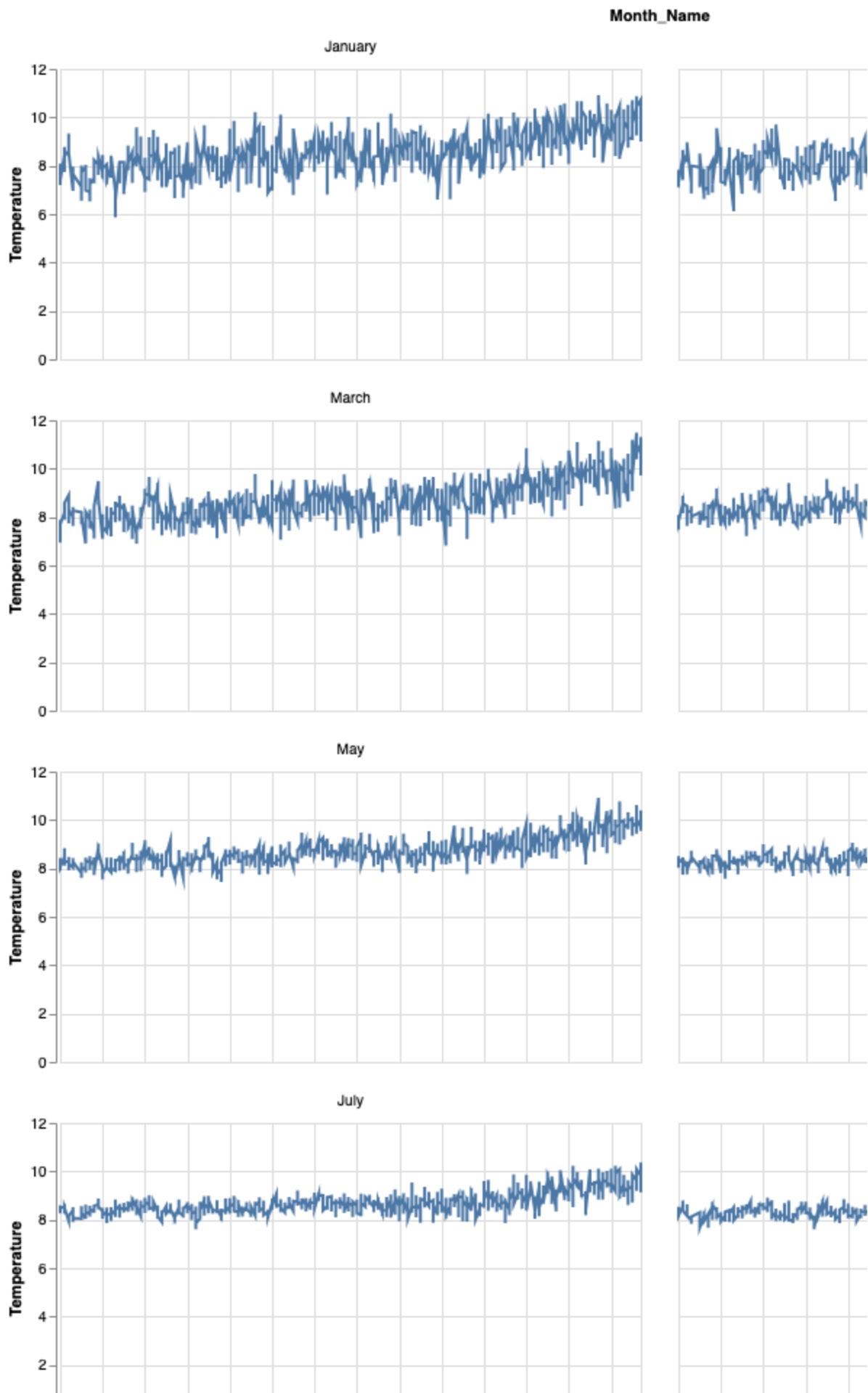
temp_plot

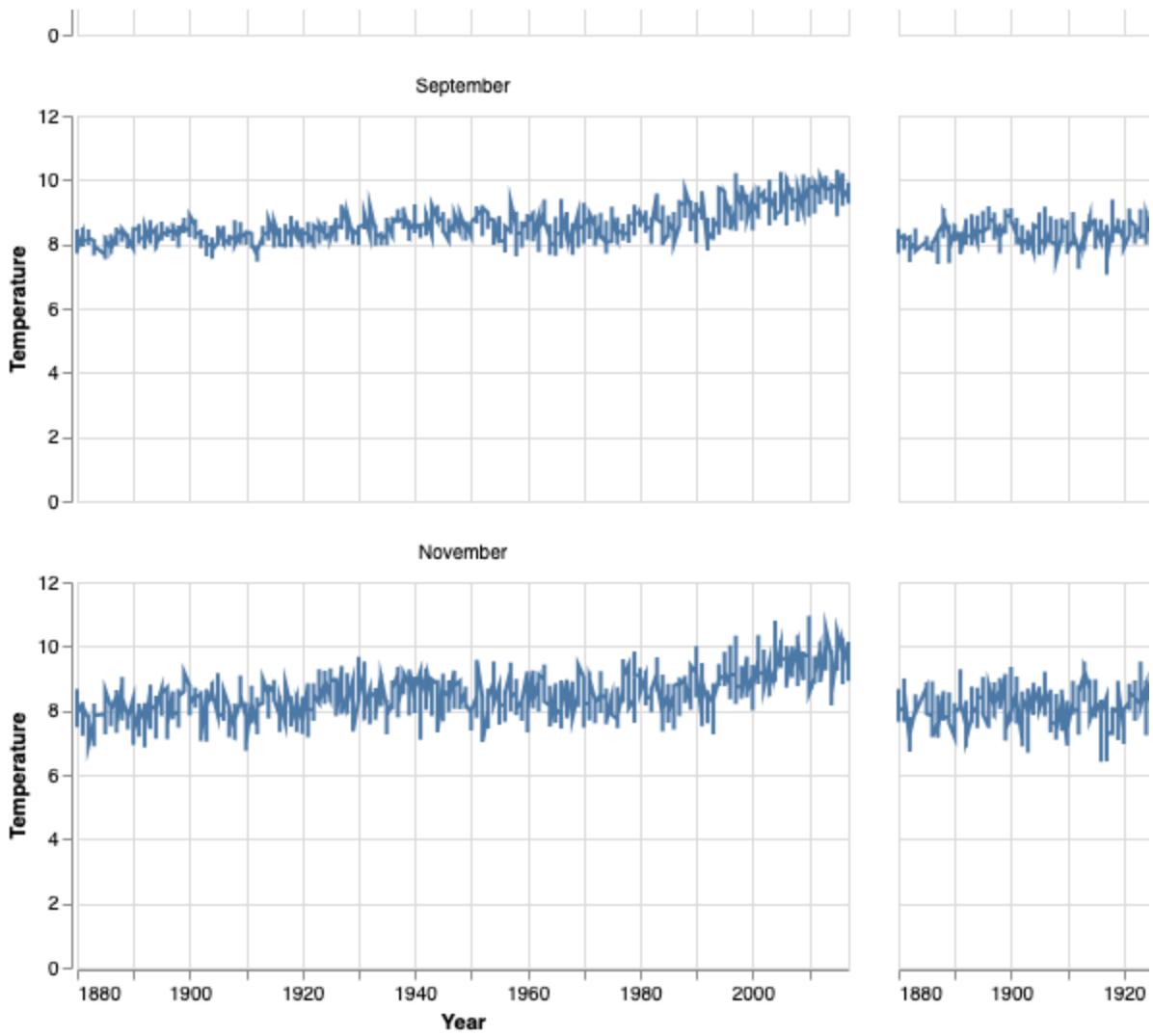
# Suffering from Overplotting, perhaps we should take the mean of the temper
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?  
return \_is\_pandas\_dataframe(obj) or isinstance(

Out[19]:





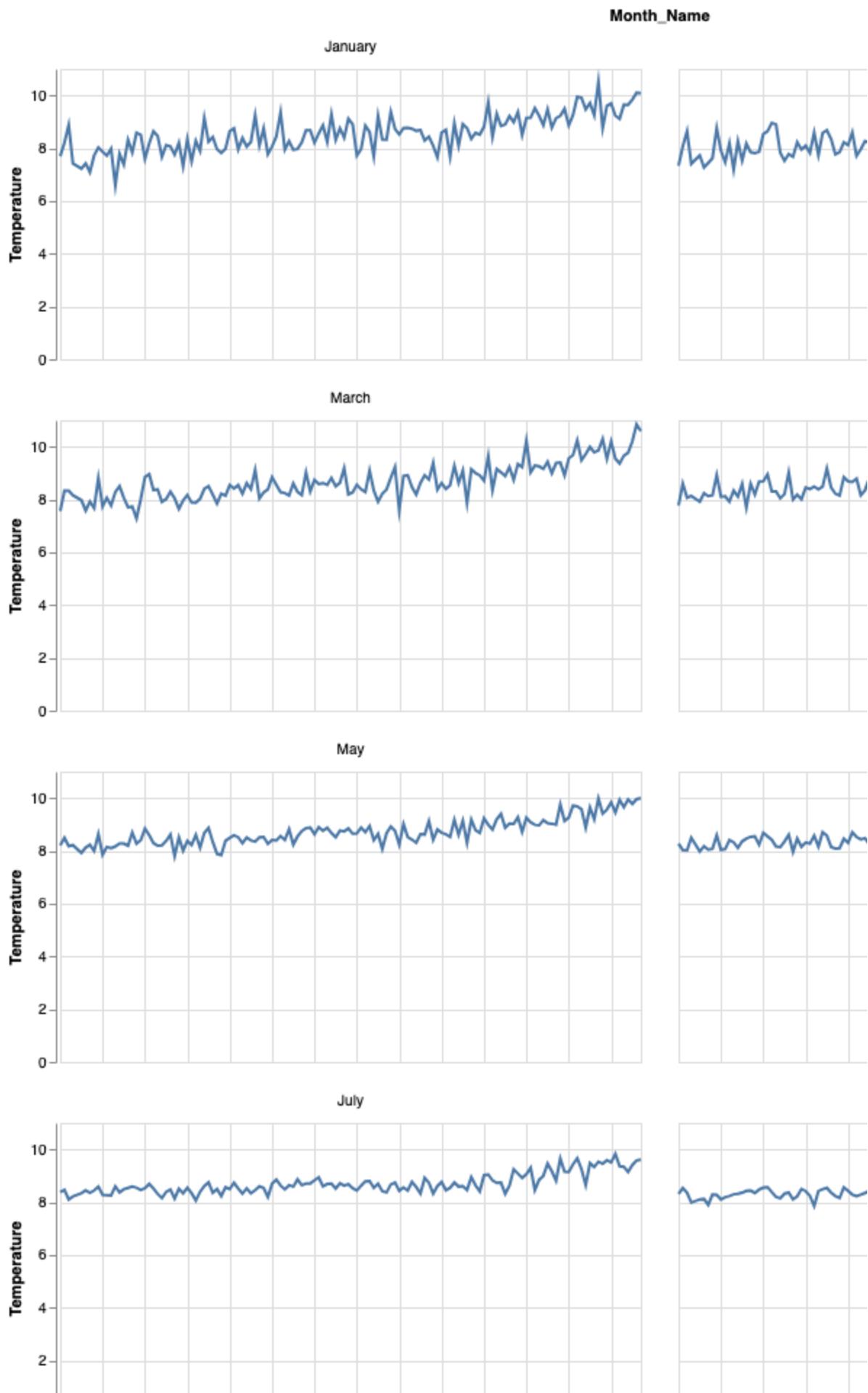
```
In [20]: mean_per_month = train_df.groupby(['Year', 'Month_Name'])['Temperature'].mean()

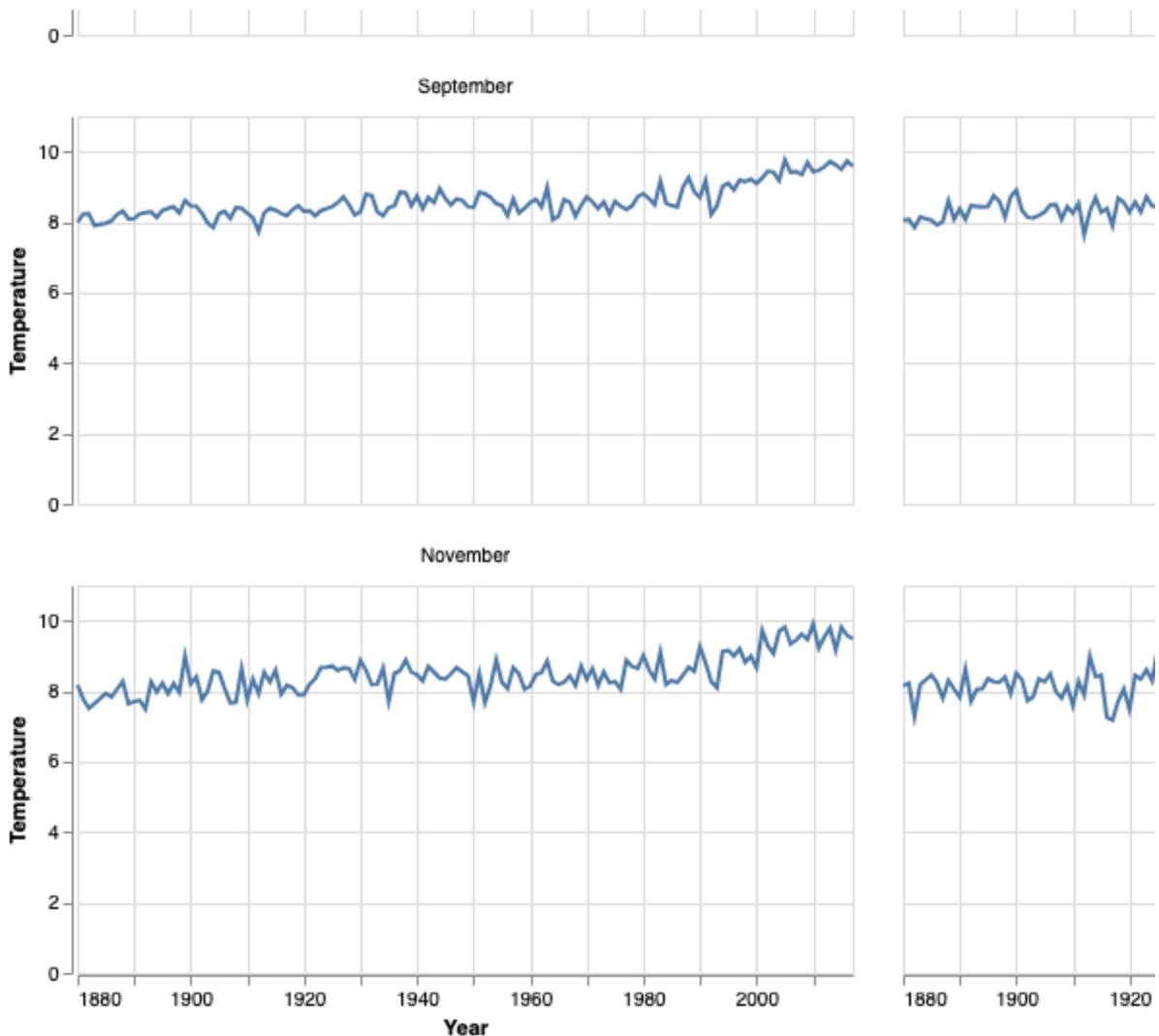
temp_plot = alt.Chart(mean_per_month).mark_line().encode(
    x = 'Year:T',
    y = 'Temperature'
).properties(width = 400, height = 200).facet('Month_Name', columns=2)

temp_plot
```

```
/var/folders/b0/mpnsyl9j37s5ywbwy416jc0000gn/T/ipykernel_23586/1756300789.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
    mean_per_month = train_df.groupby(['Year', 'Month_Name'])['Temperature'].mean().reset_index()
/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is_pandas_dataframe`.  
  
Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?  
    return _is_pandas_dataframe(obj) or isinstance(
/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is_pandas_dataframe`.  
  
Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dataframe(df.to_native())`?  
    return _is_pandas_dataframe(obj) or isinstance(
```

Out[20]:





```
In [21]: temp_plot = alt.Chart(train_df).mark_line().encode(
    x = 'Month_Name',
    y = alt.Y('mean(Temperature)').scale(zero=False)
).properties(width = 700, height = 400)

temp_plot
```

```
/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is_pandas_dataframe`.
```

```
Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dat
aframe(df.to_native())`?
```

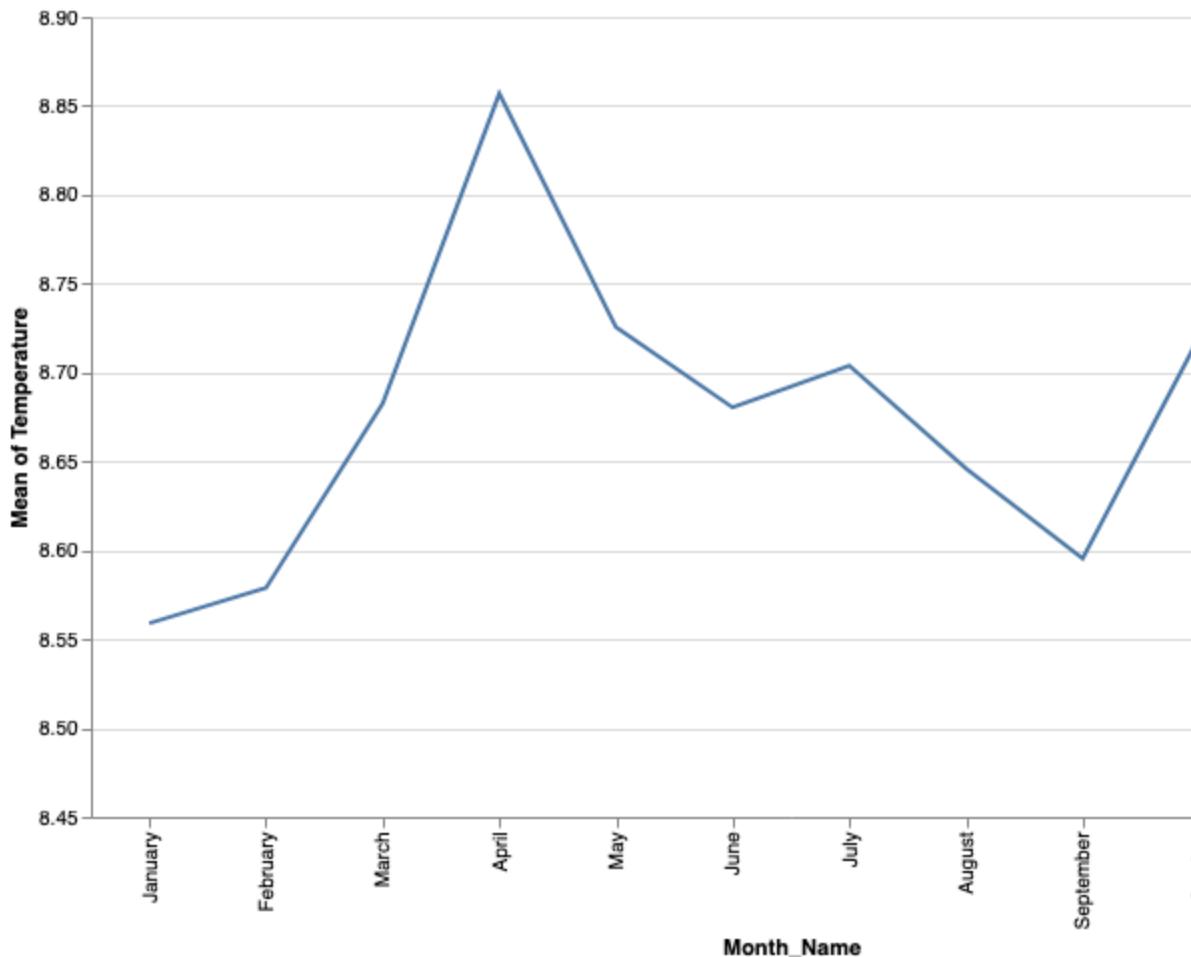
```
    return _is_pandas_dataframe(obj) or isinstance(
```

```
/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is_pandas_dataframe`.
```

```
Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dat
aframe(df.to_native())`?
```

```
    return _is_pandas_dataframe(obj) or isinstance(
```

Out[21]:



In [22]:

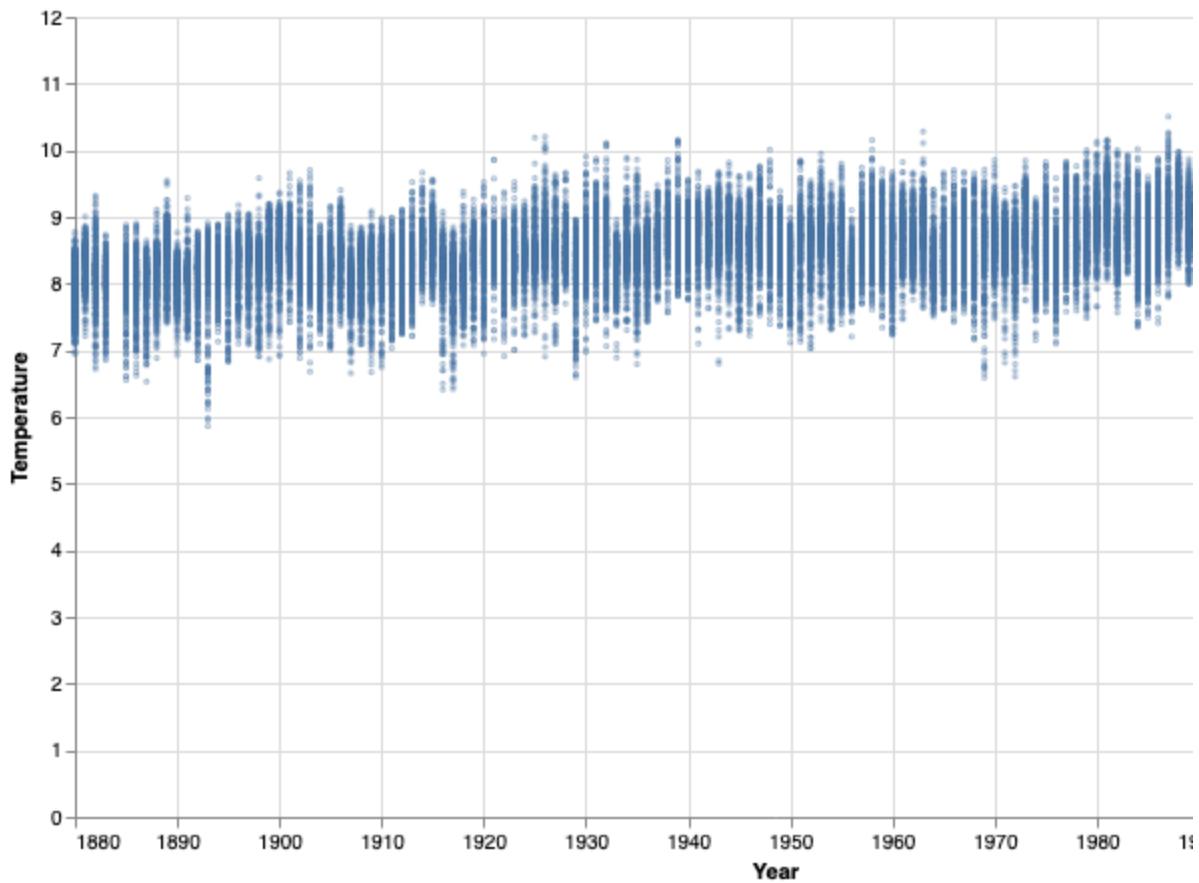
```
temp_points = alt.Chart(train_df).mark_point(opacity=0.5, size=1).encode(
    alt.X('Year:T'),
    alt.Y('Temperature:Q')
).properties(width = 700, height = 400)

temp_points
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?  
return \_is\_pandas\_dataframe(obj) or isinstance(

Out[22]:



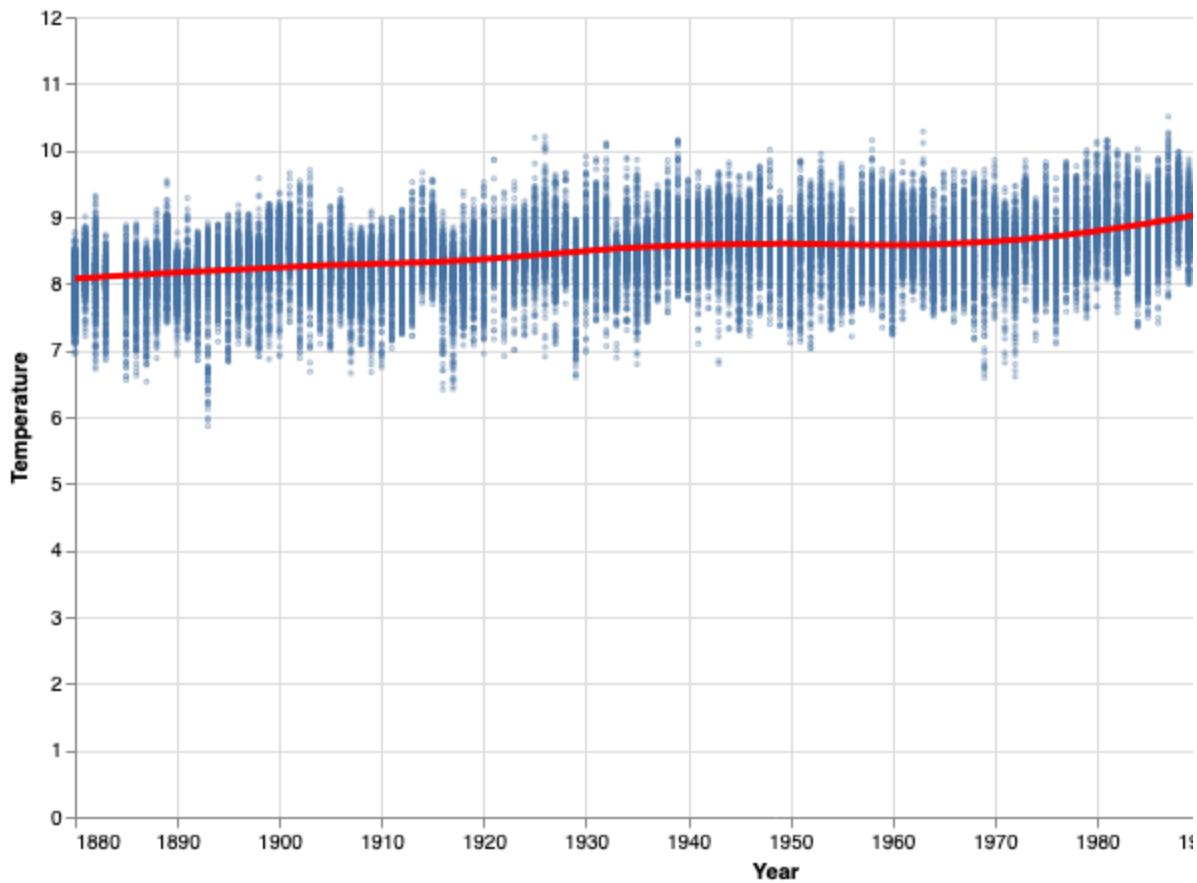
In [23]:

```
temp_points + temp_points.mark_line(size=3, color='red').transform_loess(  
    'Year',  
    'Temperature'  
)
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?  
return \_is\_pandas\_dataframe(obj) or isinstance(

Out[23]:



In [24]:

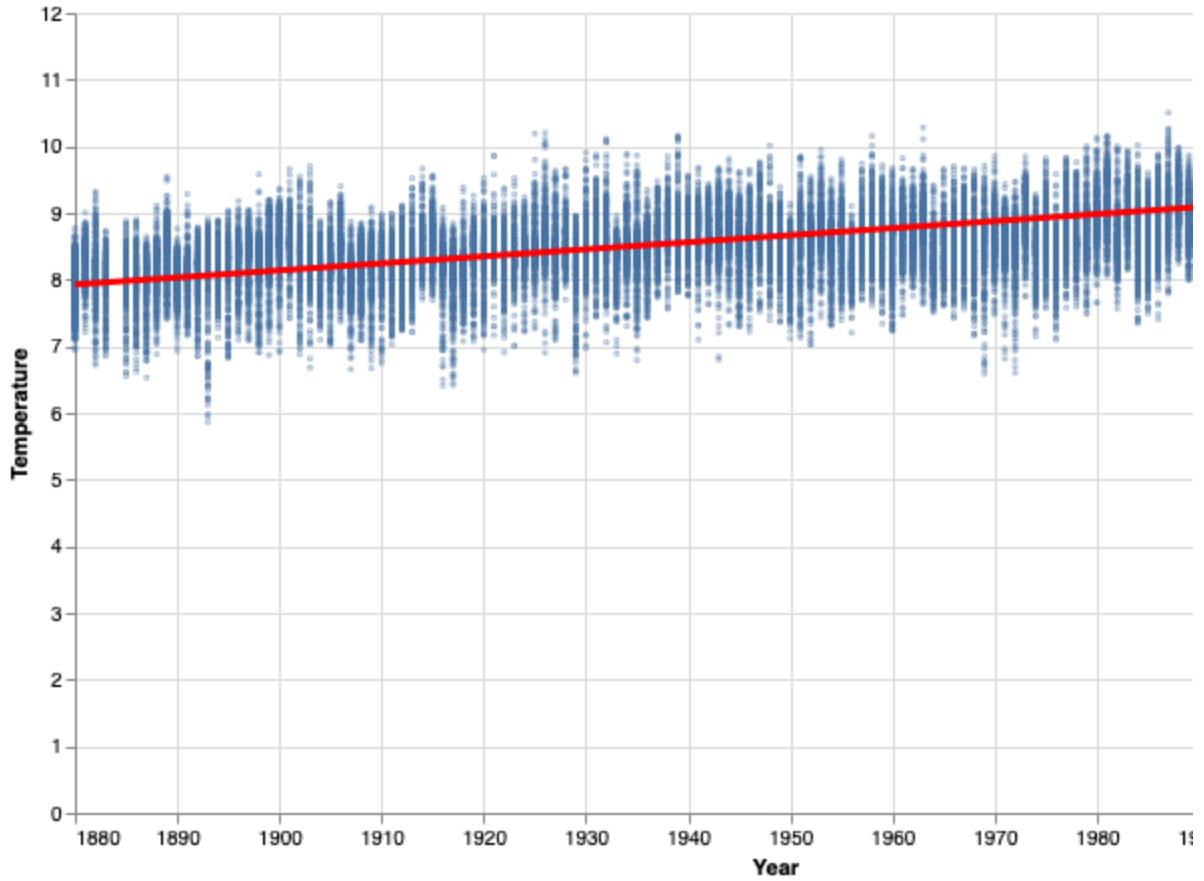
```
temp_points + temp_points.mark_line(size=3, color='red').transform_regression('Year', 'Temperature')
```

/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is\_pandas\_dataframe`.

Hint: Instead of e.g. `is\_pandas\_dataframe(df)`, did you mean `is\_pandas\_dataframe(df.to\_native())`?

return \_is\_pandas\_dataframe(obj) or isinstance(

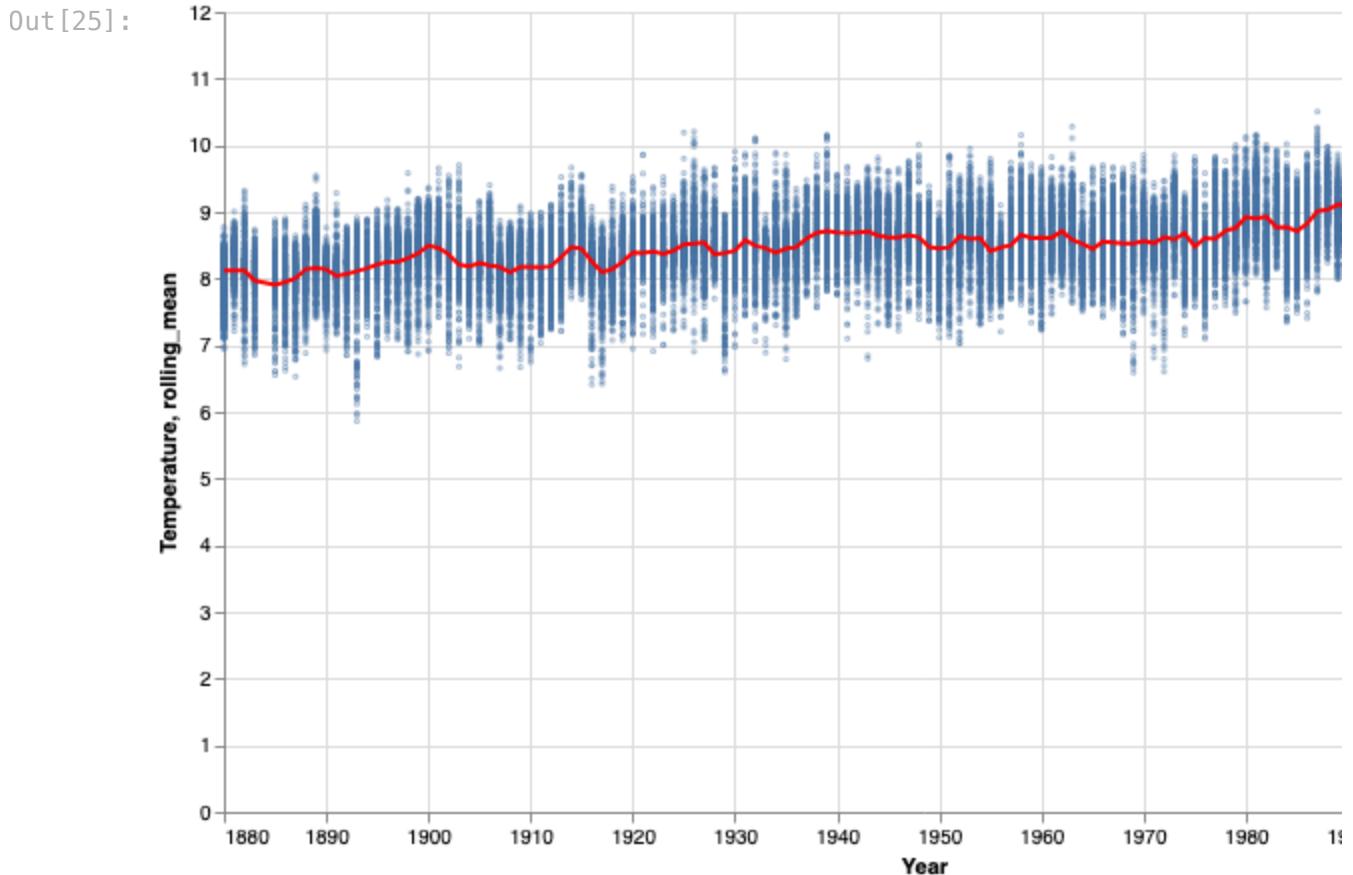
Out[24]:



In [25]:

```
# 30-day rolling average example adapted from source documentation below:  
# Adapated from 5331 Lecture 5 Notes  
# https://altair-viz.github.io/gallery/scatter_with_rolling_mean.html  
  
mean_per_year = train_df.groupby(['Year'])['Temperature'].mean().reset_index()  
  
roll_line = alt.Chart(mean_per_year).mark_line(  
    color='red',  
    size=2  
)  
.transform_window(  
    rolling_mean='mean(Temperature)',  
    frame=[-1, 1]  
)  
.encode(  
    x='Year:T',  
    y='rolling_mean:Q'  
)  
  
# Three-year moving/rolling average (day-based moving average not easy to pl  
# And day of year and day reset every new year or month  
# Daily averages may be too noisy anyway we can see the rolling average smooth  
temp_points+roll_line
```

```
/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is_pandas_dataframe`.  
Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dat  
aframe(df.to_native())`?  
    return _is_pandas_dataframe(obj) or isinstance(
```



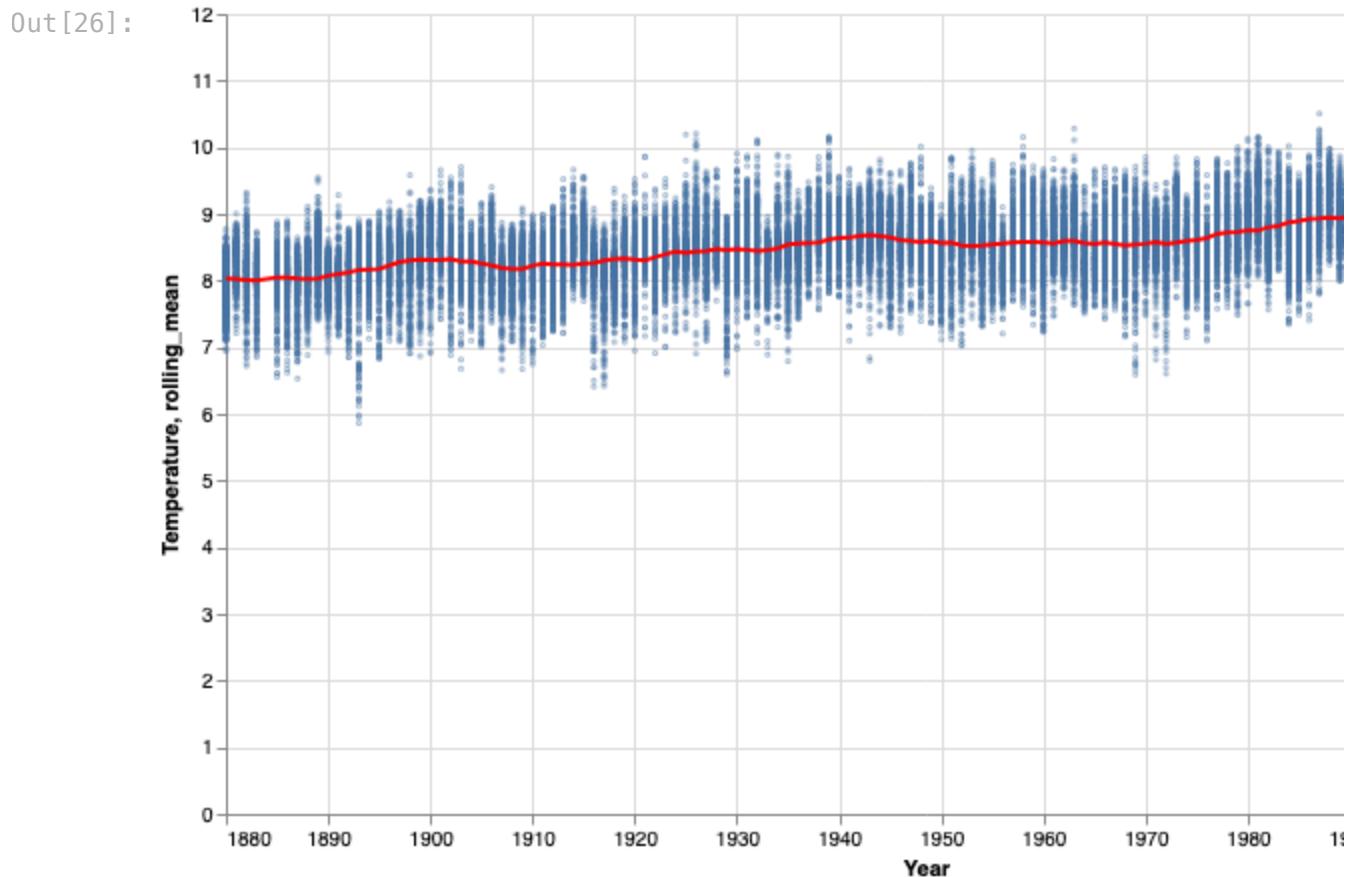
In [26]:

```
mean_per_year = train_df.groupby(['Year'])['Temperature'].mean().reset_index()

roll_line = alt.Chart(mean_per_year).mark_line(
    color='red',
    size=2
).transform_window(
    rolling_mean='mean(Temperature)',
    frame=[-5, 4]
).encode(
    x='Year:T',
    y='rolling_mean:Q'
)

# Ten-year moving/rolling average (day-based moving average not easy to plot
# And day of year and day reset every new year or month
temp_points+roll_line
```

```
/Users/jacob/miniforge3/envs/climate-env/lib/python3.11/site-packages/altair/utils/data.py:71: UserWarning: You passed a `<class 'narwhals.stable.v1.DataFrame'>` to `is_pandas_dataframe`.  
Hint: Instead of e.g. `is_pandas_dataframe(df)`, did you mean `is_pandas_dat  
aframe(df.to_native())`?  
    return _is_pandas_dataframe(obj) or isinstance(
```



## Knitting commands

```
jupyter nbconvert --to webpdf initial_eda.ipynb
```

```
jupyter nbconvert --to html initial_eda.ipynb
```